Proposing a High-Level Description for the Mathematical Structure for Fintech Project

Barış Bostancı Ege Ercan

Abstract

Our primary goal is to catch intricate patterns present in the financial markets data, allowing investors to model and control their risks. There are plenty of algorithms that can achieve it, but as a starting point, we propose trying machine learning algorithms such as Physics Informed Neural Networks (PINNs). We will provide some arguments throughout this paper as to why should we use such model, with some relevant examples we find useful.

Contents

1 First Look at PINNs **3**
1.1 Formal Structure of the Model 4
1.2 Capturing Nonlinearities - Magic Happens 4

2 Training The Model With Real World Data - Supervised Learning **6**
2.1 Hypothesis Function, Neural Model and the Loss Function 6
2.2 Optimization Over Arbitrarily Large Dimensional Spaces 7
2.3 Gradient Descent Algorithm 9
2.4 Gradient Descent on Neural Nets 11

3 Tensor Calculus **11**
3.1 Can We Generalize Further? How? by Using Tensors 13
3.1.1 Change of Basis, Dual Space of a Vector Space & Einstein Summation Convention 14
3.2 Tensor Fields, Differentiating Tensors, Metric Tensor and Covariant Derivative . 17
3.3 Matrix Calculus as a Special Case of Tensor Calculus 21
3.4 Backpropagation in its Most General Form 21

4 News Reading and Sentiment Analysis **22**

5 Probability and Stochastic Processes **22**
5.1 Defining Fundamental Concepts 22
5.2 Expectation, Variance, Multivariable Distributions, Conditional Probabilities, Independence and Bayes' Rule and Characteristic Function 26
5.3 Calculus of Probabilities 30
5.4 Markov Processes 30
5.5 Brownian Motion and Diffusion Process 30
5.5.1 Ito's Lemma 30

5.6 Fokker-Plank Equation 30
5.7 Black-Scholes Equation and Few words on Financial Models 30
5.8 Analyzing Financial Markets and Statistical Mechanics 30

6 Neural Networks Architectures **30**
6.1 Convolutional Neural Networks 30
6.2 Physics Informed Neural Networks 30
6.3 LSTM architecture and Capturing Time Series Data 30

7 General Statistical Tools **30**
7.1 Hidden Markov Models 30
7.2 Monte Carlo Methods 30
7.3 Random Forests 30

8 Implementing Our Algorithm **30**
8.1 Discussing our priorities 30
8.2 Full Algorithm 30

1 First Look at PINNs

To see why using PINNs might be a good idea to employ in financial markets, let me describe their working principles quickly. First of all they are special class of more broad category of algorithms known as neural networks. The whole idea of neural network architecture is basically to learn from training data. The events in real world can effect each other in every possible way imaginable, these effects can be highly non-linear, this non-linearity can be captured by neural nets*. The classical architecture consists of layers of nodes, also called neurons. These nodes can be connected to each other in quite intricate manner, but usually they are structured as shown in the figure below

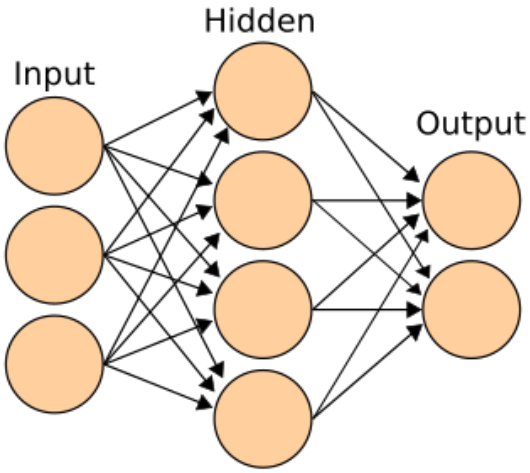

Figure 1: Basic structure of Feedforward Neural Nets

Now, mathematically speaking, we want to approximate a function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$. $\mathcal{F}$ eats n-dimensional input vector, and spits out an m-dimensional output vector. For instance, input space -domain- of this function can be though of as set of all 28x28 images (a 28x28 image contains 784 pixels, each pixel is a triple (r,g,b) $\in \mathbb{R}^3$, thus input space can be thought of as the set of all possible functions $\mathcal{G} : \{1, \ldots, 784\} \to \mathbb{R}^3$, which is equivalent to $\mathbb{R}^{784 \cdot 3}$, 2352-dimensional euclidean space) and output space can be though of as 10 dimensional vector:

$$\mathbf{y} = \begin{bmatrix} \mathcal{F}_1(x) \\ \mathcal{F}_2(x) \\ \vdots \\ \mathcal{F}_{10}(x) \end{bmatrix} \quad \text{where the N-th element of } \mathbf{y} \text{ is the probability that this image contains number N.} \tag{1}$$

-Handwritten digit recognition, basically- Now, how on earth, could someone imagine what this function would look like let alone describing it with symbolic expressions. This is where neural nets comes into place. They are trying to guess this function as accurate as they could using the neural architecture, I will shortly try to describe the algorithm that is used to achieve such a hard mission, but before that, I need to say a few words on the kind of functions that these neural architectures are allowed to construct.

*https://www.youtube.com/watch?v=pdNYw6qwuNc

1.1 Formal Structure of the Model

Here, I believe it is quite important to stick with some mathematical convention and go with it. I will describe an input as N-dimensional vector, in other words, input is $\mathbf{X} \in \mathbb{R}^N$, in our neural network, there are N nodes in its input layer, each node in this layer stores the a component of $\mathbf{X}$. I will index upmost node as σ_1^1 and lowest node σ_N^1, top index represents that we are in the 1st layer -the input layer- and down index represents which component of $\mathbf{X}$ they are storing. So basically $\sigma_i^1 = X_i$.

If you look the figure 1 above, you see that these nodes are connected, each node in the input layer is fully connected to each node in next layer, also called the hidden layer, in our notation $\sigma_i^1 \to \sigma_j^2$. These connections are also called **weights** because each connection is attached a real number W. We index those weights as follows:

> the connection from σ_i^1 to σ_j^2 is represented by $\mathrm{W}_{\mathrm{ji}}^1$. In other words, $\mathrm{W}_{\mathrm{ji}}^1$ is the weight of the connection between i-th node in the first layer to j-th node in the subsequent layer, starting from the 1st layer. (up index of $\mathrm{W}_{\mathrm{ji}}^1$ is a reminder that the connections stem from 1st layer)

Now, we can represent the connections between i-th layer and i+1-th layer as a $N_{i+1} \times N_i$ matrix $\mathbf{W}^i$, where N_i is the number of nodes present in the i-th layer. Now these connections can be thought of as encoding the linear relationship between input and output. This construction comes really handy when we want to express mathematically that how nodes in the precedent layer effects the nodes in the subsequent layer. This relationship is given in the formula below

$$\sigma^{i+1} = \mathbf{W}^i \cdot \sigma^i + \mathbf{b}^{i+1} \tag{2}$$

The above equation is a matrix equation of the form $\mathbf{Ax} + \mathbf{b} = \mathbf{c}$. To make things clear let me go over what those symbols represent more explicitly.

$$\sigma^i = \begin{bmatrix} \sigma_1^i \\ \sigma_2^i \\ \vdots \\ \sigma_{N_i}^i \end{bmatrix} \qquad \mathbf{W}^i = \begin{bmatrix} W_{11}^i & W_{12}^i & \dots & W_{1N_i}^i \\ W_{21}^i & W_{22}^i & \dots & W_{2N_i}^i \\ \vdots & \vdots & \vdots & \vdots \\ W_{N_{i+1}1}^i & W_{N_{i+1}2}^i & \dots & W_{N_{i+1}N_i}^i \end{bmatrix} \qquad \mathbf{b}^{i+1} = \begin{bmatrix} b_1^{i+1} \\ b_2^{i+1} \\ \vdots \\ b_{N_{i+1}}^{i+1} \end{bmatrix}$$

Although it looks like there are lots of shit going through, the meaning is really clear: you basically sum up all the "weighted" contributions of nodes in i-th layer to obtain values in i+1-th layer, with slight adjustment of summing $\mathbf{b}^{i+1}$ over, this is intuitively the bias of the each node in the i+1-th layer. Speaking more formally, by allowing such a construction we literally capture all possible linear relations (affine relations actually, because of the bias) between the input and output layer through iterated application of eq(2) throughout the layers between the input and output layer. All that is left for us to complete the picture is to introduce the nonlinearity into the model.

1.2 Capturing Nonlinearities - Magic Happens

Now introducing nonlinearity is a subtle thing, and is kind of ambiguous statement, what is it suppose to mean "introducing nonlinearities". Going back to the bigger picture, we are trying to quantify the relations between "things" mathematically. By even saying that, we already made two assumptions: 1.There is a quantifiable relation between "input and output", 2. This relation is mathematically expressable. Now it turns out that, with the addition of nonlinearities to the model which I will describe now, they become **Universal Approximators**, which means, if

there is some mathematical function that describes the relation between input and output, then with this model, the relation function can be approximated up to arbitrary precision.[†]

Capturing nonlinearities with our model can be achieved by introducing additional step to eq(2) that describes how nodes in successive layers effect each other. Let me describe the process and then explain what does it mean

$$\sigma^{i+1} = \phi(\mathbf{W}^i \cdot \sigma^i + \mathbf{b}^{i+1}) \tag{3}$$

Here, $\phi : \mathbb{R} \to \mathbb{R}$ is a nonlinear function called the **activation function**, it is usually chosen as one of the following functions below:

1. sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$
2. Hyperbolic tangent function: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
3. ReLU: $f(x) = \max(0, x)$
4. and million more, depending on the situation at hand. These functions usually take all the weighted sum from previous layer and squish it to interval $[-1, 1]$, this is why these nodes are called neurons, and those functions are called activation functions, if node is an active for given input its activation is positive, if it should inhibit it gets negative, if it is irrelevant it gets close to 0. Just like how the brain works.

In the equation (3) I apply the function in the point-wise manner, that is:

$$\mathbf{z}^{i+1} = \mathbf{W}^i \cdot \sigma^i + \mathbf{b}^{i+1} \tag{4}$$

$$\mathbf{z}^{i+1} = \begin{bmatrix} z_1^{i+1} \\ z_2^{i+1} \\ \vdots \\ z_{N_{i+1}}^{i+1} \end{bmatrix} \quad \phi(\mathbf{z^{i+1}}) = \begin{bmatrix} \phi(z_1^{i+1}) \\ \phi(z_2^{i+1}) \\ \vdots \\ \phi(z_{N_{i+1}}^{i+1}) \end{bmatrix}$$

Now, let me describe the process of constructing functions with a final diagram:

[†] `https://en.wikipedia.org/wiki/Universal_approximation_theorem`

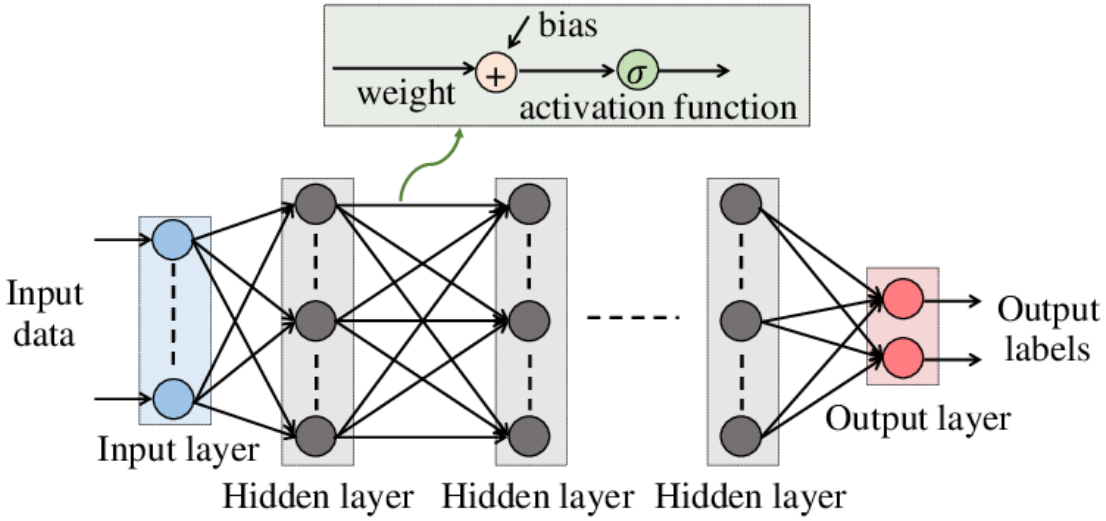

Figure 2: Summary of the Process

2 Training The Model With Real World Data - Supervised Learning

Now that we have the model, it is now, that the ingenious algorithms come into the picture. Intuitively, we want to construct an algorithm that makes our model learn from data and capture the relations between data and the output (in our case the input will be "S&P index, news index, and other things we might add as we go on with the project", and the output will be the price of the stock, or option, or another weird things we might discover as helpful.) Although the training business has lots of subtleties and convergence issues, in order not to get to verbose in this introductory proposal, I won't go into that -but we eventually need to- and just introduce a classic method known as backpropagation, and then I will dive into financial stochastic differential equations, and how we might combine them.

2.1 Hypothesis Function, Neural Model and the Loss Function

Now, in order for us stick with above mathematical convention, I need to introduce few other terms in order not to get lost in notation. The hypothesis function for a model $\mathcal{M}$ is the expressed by $\mathcal{F}_{\mathcal{M}} : \mathbb{R}^N \to \mathbb{R}^M$. This function eats inputs and yields outputs, according to weights and biases in the model. Using our convention we can also express N = N_1, M =N_k, where 1st layer is input layer, and kth layer (last layer) is output layer, assuming that our model consists of k layers.

A model $\mathcal{M}$ is basically the configuration of the neural network: the weights $\mathbf{W}^i$, biases $\mathbf{b}^{i+1}$ and number of layers, and number of nodes in each layer N_i, for $i = 0, ..., k$. By using backpropagation algorithm we adjust the model (change its weights, its biases etc.) according to the training data. But before going into that, we need to define and refine few concepts.

Using eq(3) we can construct the Hypothesis function as follows:

$$\mathcal{F}_{\mathcal{M}} = \phi(\mathbf{z}^k) \text{ where } \mathbf{z}^k = \phi(\sigma^{\mathbf{k}}) \tag{5}$$

$$\text{where } \sigma^k = \phi(\mathbf{W}^{k-1} \cdot \sigma^{k-1} + \mathbf{b}^k) \tag{6}$$

Where k is the output layer. This is a recursive definition, but I believe it is clear as to what it represents

Now it is time that we define the Loss Function, this is the cornerstone concept of machine learning. Intuitively the Loss function -or the Cost function- is a measure of how well our model $\mathcal{M}$ is predicting the training data inputs, compared to training data outputs. To be mathematically concise let us define these concepts.

Training data $\mathcal{T}$ is set of tuples $\left\{(\mathbf{X}^i, \mathbf{Y}^i)\right\}_{i=1}^{S}$ where $\mathbf{X}^i \in \mathbb{R}^N$ (the input) and $\mathbf{Y}^i \in \mathbb{R}^M$ (the output) and S is the number of such training tuples we have. So it is basically the S-many examples we have where we know what the input $\mathbf{X}^i$ and the corresponding output $\mathbf{Y}^i$ is.

Now, what would be the most intuitive way of measuring how well our model $\mathcal{M}$ is performing with respect to the training data. It turns out that there are lots of way to define such a measure, but for the same reasons as I stated above, I cut short and introduce the most classical one. The Loss Function $\mathcal{L}$ is given by the expression below

$$\mathcal{L}(\mathcal{M}) = \frac{1}{S}\sum_{i=1}^{S} ||\mathcal{F}_{\mathcal{M}}(\mathbf{X}^i) - \mathbf{Y}^i||^2 \quad \text{Where } (\mathbf{X}^i, \mathbf{Y}^i) \in \mathcal{T} \tag{7}$$

Where $||\mathbf{x} - \mathbf{y}||^2 = \sum_{i=1}^{M}(x_i - y_i)^2$ the square of the euclidean distance for $\mathbb{R}^M$, Loss function is basically measuring how far our model is predicting from what it should be predicting. Averaging this discrepancy for all S-many examples gives us an overall measure of how well our model $\mathcal{M}$ is with respect to given training data $\mathcal{T}$

Now, the notation might seem a bit odd, and the definition might seem perplexing, but it is not. Loss function is a function of our model $\mathcal{M}$, what kind of space is it? It is the euclidean space whose dimension is sum of all parameters present in the model, that is, number of weights + number of biases. For instance, every possible model $\mathcal{M}$ for the neural net depicted in the figure (1) can be thought of as a point of $\mathbb{R}^{26}$ (number of weights: 3x4 + 4x2 = 20, number of biases: 4 + 2 = 6. In total 26 parameters that controls the space). That is for every possible configuration of values of weights, and biases ("trainable parameters") Loss function gives us a real number. The whole idea of backpropagation is to figure a way out to minimize the Loss function over arbitrarily large dimensional spaces.

2.2 Optimization Over Arbitrarily Large Dimensional Spaces

(bkz. Yüzüp Yüzüp Kuyruğuna Gelememek, bkz. Unreasonable Effectiveness of Mathematics) There is a saying "Immature optimization is the root of all evil.", well it is, but before that, what is optimization, I need to dissolve all ambiguity in order to keep things tractable. Imagine a function $f : \mathbb{R} \to \mathbb{R}$. We can define two notions of optimization, global optimization and local optimization. Global optimization is the process of finding the global minimum or maximum of a given function, local optimization defined similarly. The most effective tool we possess, that

might help us to achieve such a goal is... well, calculus. In single variable calculus it is quite simple to find the values for which function is optimized, you simply seek the points for which $\frac{df}{dx} = 0$, and second derivative is tell you whether you find the maxima or minima. However as dimension grow, (even up to infinity) the process become much more cumbersome, but still manageable for time to time. Imagine a function $f(x, y) = z$ whose graph is given above in figure(3). The method for finding local minima/maxima is now changed. One needs to consider the points for which gradient of the function f vanishes, i.e. $\nabla f|_{\mathbf{x}=(a_1,a_2)} = 0$. The gradient of a function $g : \mathbb{R}^n \to \mathbb{R}$ is given by

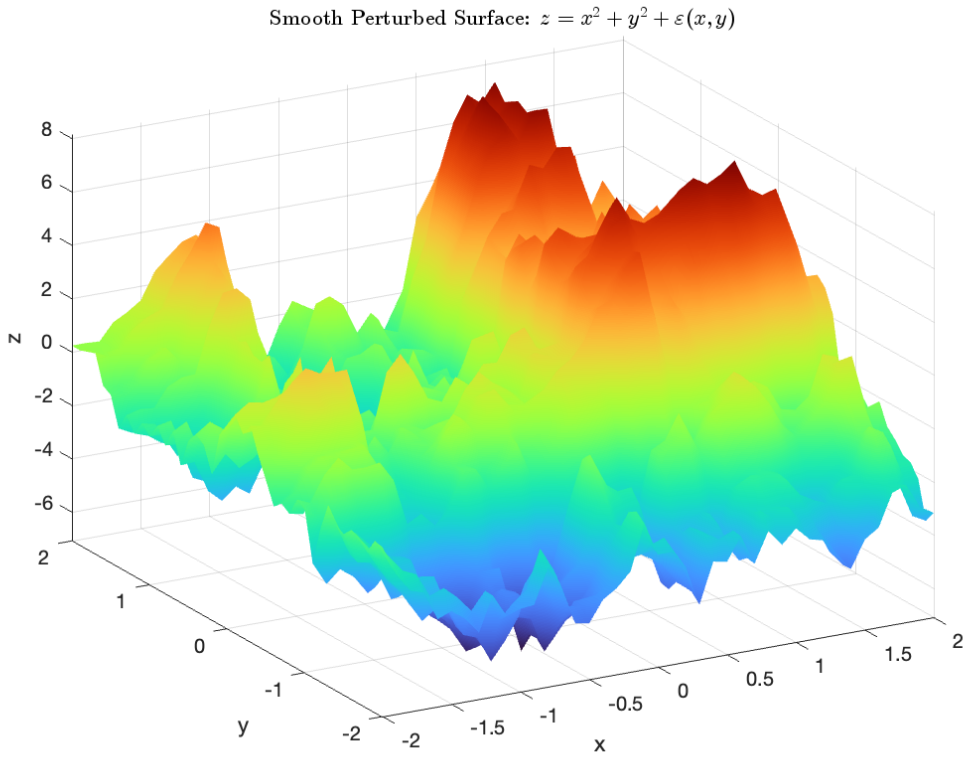

Figure 3: Graph of some weird function f(x,y) = z

$$\nabla g|_{\mathbf{x}=(a_1,\ldots,a_n)} = \left[\frac{\partial g}{\partial x_1}|_{\mathbf{x}} \quad \frac{\partial g}{\partial x_2}|_{\mathbf{x}} \quad \cdots \quad \frac{\partial g}{\partial x_n}|_{\mathbf{x}}\right] \tag{8}$$

Where $\frac{\partial g}{\partial x_i}|_{\mathbf{x}}$ denotes the partial derivative of g evaluated at the point $\mathbf{x}$. Now at first glance this definition of minima/maxima might not make too much sense, but if you realize that the minima/maxima points of the function f are those that has its first order approximation equals to 0 function, then you might see why this is the case. Derivative, when generalized to multidimensional settings, defined as the "best linear approximation" of the function[‡], and such is given by the gradient (more generally jacobian of a function), you can think gradient as the

[‡]https://en.wikipedia.org/wiki/Total_derivative

best linear approximation of the function, or for two dimensional case, the tangent plane that sits on the graph, but this analogy is not reproductive in dimensions more than 2.

Now, going back to the optimization of loss function $\mathcal{L}$, our goal is to optimize this function to its minima. You can imagine the graph of a Loss function as somewhat similar to figure(3) it has lots of bumps, and irregular, and we want to find a configuration of weights and biases so that the model $\mathcal{M}$ with these parameters minimizes the Loss function, and we usually need to perform this optimization over arbitrarily large dimensions such as $\mathbb{R}^{1000000}$.

2.3 Gradient Descent Algorithm

Gradient descent algorithm is an iterative method for optimizing differentiable functions. The reason why this algorithm is called the gradient descent is that it uses the fact that the gradient of the function gives the direction of **steepest ascend** of the function. I will unpack this idea with both formal and intuitive approach. Formal proof of this statement is given as follows: Let $f : \mathbb{R}^n \to \mathbb{R}$ be a totally differentiable function, then, for a given vector $\mathbf{u} \in \mathbb{R}^n$, the first order approximation $T^1_{f_\mathbf{x}}$ of f at $\mathbf{x}$ is given by:

$$T^1_{f_\mathbf{x}}(\mathbf{x}+\mathbf{u}) = f(\mathbf{x}) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}|_\mathbf{x} \cdot u_i \tag{9}$$

Same thing can also be expressed in terms of matrices as

$$T^1_{f_\mathbf{x}}(\mathbf{x}+\mathbf{u}) = f(\mathbf{x}) + \nabla \mathbf{f}|_\mathbf{x} \cdot \mathbf{u} \tag{10}$$

(bkz. Figure 4) To verify that, let us calculate this matrix product

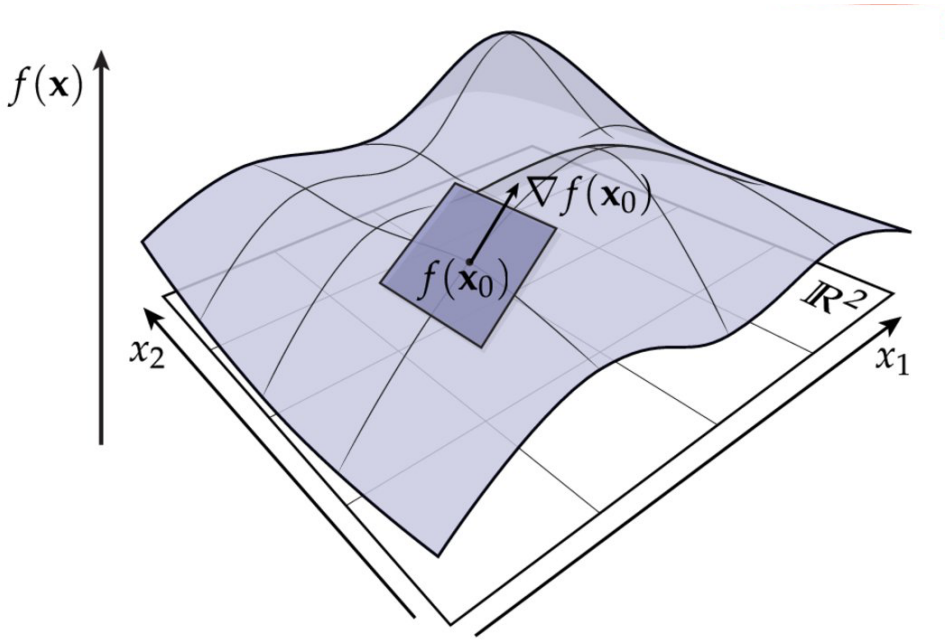

Figure 4: Gradient of the function

$$\nabla \mathbf{f}|_{\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1}|_{\mathbf{x}} & \frac{\partial f}{\partial x_2}|_{\mathbf{x}} & \cdots & \frac{\partial f}{\partial x_n}|_{\mathbf{x}} \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial f}{\partial x_1}|_{\mathbf{x}} & \frac{\partial f}{\partial x_2}|_{\mathbf{x}} & \cdots & \frac{\partial f}{\partial x_n}|_{\mathbf{x}} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}|_{\mathbf{x}} \cdot u_i \tag{11}$$

Now, expressing these operation using linear algebra helps us to keep things in order and as compact as possible. We want to show that the vector $(\nabla \mathbf{f}|_{\mathbf{x}})^T$ (transpose of the gradient is called the gradient vector) is the direction for which the function is increasing the most. It is suffices, then, to show that first order approximation of f is maximized when $\frac{\mathbf{u}}{||\mathbf{u}||} = \frac{\nabla \mathbf{f}}{||\nabla \mathbf{f}||}$ (this is mathematically accurate way to say that two vectors have the same direction). Looking at right hand side of eq(9), one can realize that the expression $\nabla \mathbf{f}|_{\mathbf{x}} \cdot \mathbf{u}$ is equivalent to dot product of two vectors $(\nabla \mathbf{f})^T$ and $\mathbf{u}$. Then it is a simple mathematical fact that the dot product of two vectors are maximized when they are in the same direction. This completes the proof. □

Intuitively, gradient vector $(\nabla \mathbf{f})^T$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ can be understood as vector a field $(\nabla \mathbf{f})^T : \mathbb{R}^n \to \mathbb{R}^n$ whose value at $\mathbf{x}$ is the vector $(\nabla \mathbf{f})^T(\mathbf{x}) = (\nabla \mathbf{f}|_{\mathbf{x}})^T$ that is perpendicular to level line of the function f that pass through $\mathbf{x}$ and whose magnitude is proportional to how fast function changes around that point, figure 5 illustrates the point.

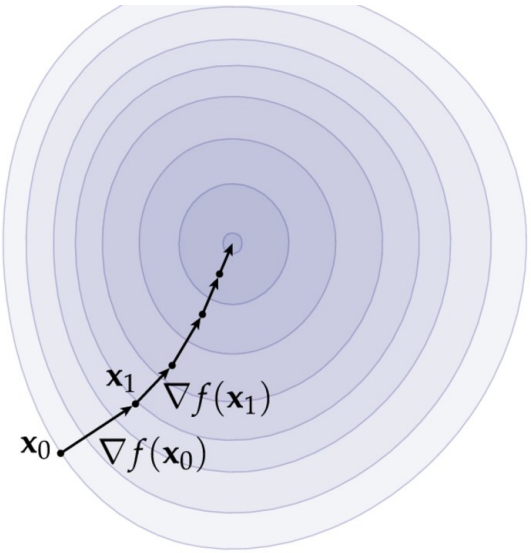

Figure 5: Gradient as a vector field

Now, I believe we are ready for full description of the algorithm

```
Input: A function f from Rn to R, randomly selected initial point k in Rn,
       r (step size), epsilon (exit error)

Base Step: i = 0, k_0 = k


Iterate: k_(i+1) = k_i -  r grad(f(k_i))


Exit: If ||f(k_(i+1)) - f(k_i)|| > epsilon, k = k_(i+1)


Return k.
```

2.4 Gradient Descent on Neural Nets

Above is the general algorithm for gradient descent on random functions, now let us focus our attention to loss function -cost function- $\mathcal{L}(\mathcal{M})$. Training a neural network is basically another way of saying **optimizing the loss function of a neural net for a given training data**. The domain of the loss function is set of all models for the given neural network. The dimension of that space is the sum of all trainable parameters of the neural network, i.e. weights and biases. Every point in that space can be thought of as some model $\mathcal{M}$ for which the loss function $\mathcal{L}(\mathcal{M})$ has a corresponding value. It is the goal of this section to analyze how gradient descent algorithm would change the model, and whether it always works; moreover, if it works, how long does it take for the algorithm to converge on a minima model $\mathcal{M}$. Now intuitively the process is not too hard to conceive, but in order for us to really understand how the process work rigorously, we need to express everything mathematically, so that we can implement those ideas into algorithms. For that reason I must introduce another mathematical convention, a mathematical framework in which we operate. Therefore, my fellow scholar, I present to you the Tensor Calculus...

3 Tensor Calculus

Dealing with neural networks requires doing calculus on multi-dimensional objects. This might sound ambiguous, and it is ambiguous right now. The whole idea of this chapter is to provide a brief introduction to make this notion precise. Matrices are multi-dimensional objects, there are a few slightly different ways of understanding matrices. The most rigorous one would be to say that matrices are representations of linear operators $\mathbf{T} : \mathbf{V} \to \mathbf{W}$ that maps $\mathbf{v} \in \mathbf{V}$ to $\mathbf{T}(\mathbf{v}) \in \mathbf{W}$ such that $\mathbf{T}(\mathbf{v} + \mathbf{u}) = \mathbf{T}(\mathbf{v}) + \mathbf{T}(\mathbf{u})$ and $\mathbf{T}(c\mathbf{v}) = c\mathbf{T}(\mathbf{v})$. Now it turns out that, for finite dimensional vector spaces, all possible linear transformations between finite dimensional vector spaces can be represented by matrices. Formally

With the matrix multiplication $\cdot_{\mathbb{M}}$ and matrix addition $+_{\mathbb{M}}$ the set $\mathbb{M}$ becomes a group $(\mathbb{M}(n, m), +_{\mathbb{M}}, \cdot_{\mathbb{M}})$; every element $\mathbf{M} \in \mathbb{M}$ represent a linear operator $\mathbf{T}$ between n-dimensional vector space $\mathbf{V}$ and m-dimensional vector space $\mathbf{W}$, the connection is $\mathbf{T}(\mathbf{v}) = \mathbf{M} \cdot \mathbf{v}$ and, when n = m, the composition of two linear operators $(\mathbf{S} \circ \mathbf{T})(\mathbf{v}) = (\mathbf{N} \cdot \mathbf{M}) \cdot \mathbf{v}$. There is a correspondence between composition and matrix multiplication

Now, when we generalize single variable calculus to multi-variable calculus, we also generalize the notion of the derivative to something that is more profound:

For a given differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$ we define the **differential, $\mathbf{df_p}$, of a function f at a point $\mathbf{p} \in \mathbb{R}^n$** as a linear operator $\mathbf{df_p} : \mathbb{R}^n \to \mathbb{R}^m$:

$$\mathbf{df_p}(\mathbf{u}) = \mathbf{J_p f} \cdot \mathbf{u} \tag{12}$$

Where $\mathbf{J_p f}$ is called the **Jacobian of f at point $\mathbf{p}$**, and it is defined similarly to a gradient (actually Jacobian is just a generalization of the gradient operator, only difference between them is that, we define the gradient for **real valued functions** whereas Jacobian is defined for **vector valued function**, so it is just a convention, and the Jacobian of real valued functions is just the gradient of that function.)

$$\mathbf{J_p f} = \begin{bmatrix} \nabla \mathbf{f^1}|_{\mathbf{p}} \\ \nabla \mathbf{f^2}|_{\mathbf{p}} \\ \vdots \\ \nabla \mathbf{f^m}|_{\mathbf{p}} \end{bmatrix} \tag{13}$$

Where $f^i : \mathbb{R}^n \to \mathbb{R}$ is the i-th component function of the f, more explicitly:

$$f(\mathbf{p}) = \big(f^1(\mathbf{p}), \ldots, f^m(\mathbf{p})\big) \tag{14}$$

So the full form of the Jacobian can be written for a function $f : \mathbb{R}^n \to \mathbb{R}^m$ explicitly as:

$$\mathbf{J_p f} = \begin{bmatrix} \frac{\partial f^1}{\partial x_1}|_{\mathbf{p}} & \frac{\partial f^1}{\partial x_2}|_{\mathbf{p}} & \cdots & \frac{\partial f^1}{\partial x_n}|_{\mathbf{p}} \\ \frac{\partial f^2}{\partial x_1}|_{\mathbf{p}} & \frac{\partial f^2}{\partial x_2}|_{\mathbf{p}} & \cdots & \frac{\partial f^2}{\partial x_n}|_{\mathbf{p}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f^m}{\partial x_1}|_{\mathbf{p}} & \frac{\partial f^m}{\partial x_2}|_{\mathbf{p}} & \cdots & \frac{\partial f^m}{\partial x_n}|_{\mathbf{p}} \end{bmatrix} \tag{15}$$

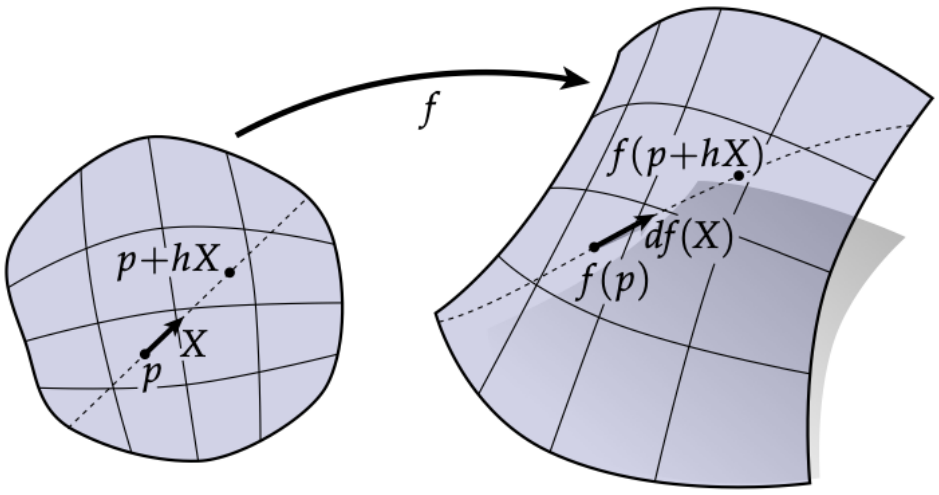

Figure 6: Generalized notion of derivative

Figure 6 beautifully demonstrates the concept of differential, imagine a function $f : \mathbb{R}^2 \to \mathbb{R}^3$, and imagine we are at the point $\mathbf{p} \in \mathbb{R}^2$, we want to take a small step from $\mathbf{p}$ along the direction $h\mathbf{X} \in \mathbb{R}^2$. -where h is an arbitrarily small small number (X is scaled by factor of h)- Then we ask the critical question: How does the function f changes when we take this step. That is can we find $f(\mathbf{p}+h\mathbf{X}) - f(\mathbf{p})$ as $\lim_{h\to 0}$? The answer is yes, and the change in the function is given by the differential of f.

$$f(\mathbf{p} + h\mathbf{X}) - f(\mathbf{p}) = \mathbf{df_p}(hX) \tag{16}$$

Or equivalently,

$$f(\mathbf{p} + h\mathbf{X}) - f(\mathbf{p}) = \mathbf{J_p f} \cdot h\mathbf{X} \tag{17}$$

This is the generalization of the notion of derivative of a function $h : \mathbb{R} \to \mathbb{R}$ in single variable calculus, eq(17) is the general version of,

$$h(x + \Delta x) - h(x) = \frac{dh}{dx}\Delta x \tag{18}$$

Looking at the similarities between these two expressions, we can say that $\mathbf{J_x f}$ is the generalized version of derivative for functions between real vector spaces (actually, more generally, between two arbitrary inner product spaces).

3.1 Can We Generalize Further? How? by Using Tensors

(Bu bölümü kısaca anlatmak biraz zor, o yüzden istersen section 3.2'ye atlayabilirsin)
Understanding tensors is somewhat difficult as there are not any clear visual representation of these objects. Tensors are natural extension of matrices, they are mathematical objects that stores multi-dimensional data, similar to the matrices. To understand where they come from, follow the analogy given below: Numbers are rank-0 tensors, vectors are rank-1 tensors, matrices are rank-2 tensors, ... Try to imagine what would a rank-3 tensor would look like:

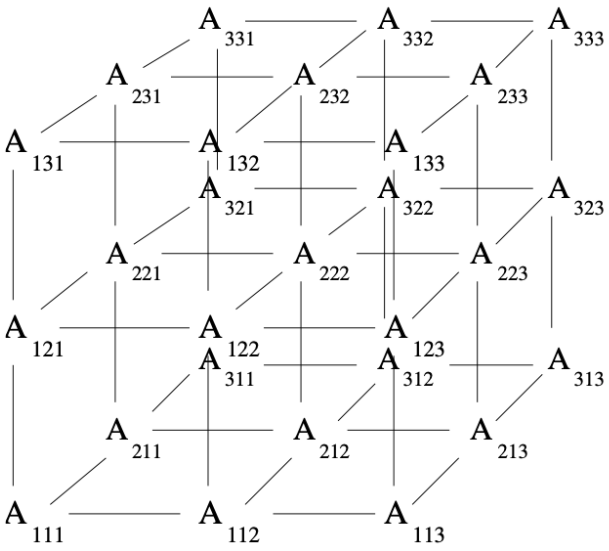

Figure 7: Rank 3 tensor

Can we formalize the tensors just as we formalized matrices by defining a matrix product and matrix addition? Is tensors also representation of linear maps just like matrices? Answers to both of these question are yes. Tensors represent multi-linear maps, for instance, a special case of multi-linear maps are bilinear maps:

$$\mathbf{B} : \mathbf{V} \times \mathbf{W} \to \mathbf{Z} \tag{19}$$

where $\mathbf{V}, \mathbf{W}, \mathbf{Z}$ are vector spaces, $\mathbf{B}$ is called a bilinear map if:

1. $\mathbf{B}$ is linear in each of its entries: $\mathbf{B}(\mathbf{v}+\mathbf{u}, \mathbf{x}) = \mathbf{B}(\mathbf{v}, \mathbf{x}) + \mathbf{B}(\mathbf{u}, \mathbf{x})$ and $\mathbf{B}(\mathbf{v}, \mathbf{x}+\mathbf{y}) = \mathbf{B}(\mathbf{v}, \mathbf{x}) + \mathbf{B}(\mathbf{v}, \mathbf{y})$
2. for any $c \in \mathbb{R}$ $\mathbf{B}(c\mathbf{u}, \mathbf{v}) = c\mathbf{B}(\mathbf{u}, \mathbf{v})$ and $\mathbf{B}(\mathbf{u}, c\mathbf{v}) = c\mathbf{B}(\mathbf{u}, \mathbf{v})$

One can extend this definition to tri-linear, quad-linear, ..., n-linear and so on, using the similar conditions above. Consider the special class of bilinear maps where

$$\mathbf{B} : \mathbf{V} \times \mathbf{V} \to \mathbb{R} \tag{20}$$

The tensorial representation of $\mathbf{B}$ can be constructed in the similar manner to the construction of the matrix representation of linear maps. Given a basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ for $\mathbf{V}$ we define:

$$B_{ij} = \mathbf{B}(\mathbf{v_i}, \mathbf{v_j}) \tag{21}$$

This construction gives us a rank-2 tensor (a matrix). The tensorial representation of $\mathbf{B}$ with respect to chosen basis for $\mathbf{V}$ is given below:

$$\mathbf{B} = B_{ij}(\mathbf{v^i} \otimes \mathbf{v^j}) \tag{22}$$

The symbol $\otimes$ is called a tensor product, and it represents the basis element of the rank-2 n^2 dimensional tensor space. Changing the coefficients B_{ij} one can construct any bilinear map $\mathbf{B} : \mathbf{V} \times \mathbf{V} \to \mathbb{R}$.

Now, I want to direct your attention to some detail in eq(22). You can see that I denoted $\mathbf{v}^i$ with up index whereas I denoted the basis elements of $\mathbf{V}$ with down index $\mathbf{v}_i$.That was conscious, and I have a good reason for doing it. Let me explain:

3.1.1 Change of Basis, Dual Space of a Vector Space & Einstein Summation Convention

It is possible to label vectors as contravariant or covariant depending on how their components change under change of basis. We know, for instance, that a change of basis can be represented by an invertible $n \times n$ matrix $\mathbf{M}$ for a given n-dimensional vector space $\mathbf{V}$. Given two bases $\mathcal{B}$ and $\tilde{\mathcal{B}}$ for $\mathbf{V}$ and a transformation between them $\mathbf{M}_{\mathcal{B}\to\tilde{\mathcal{B}}}$ the components of a vector $\mathbf{u} \in \mathbf{U}$ changes as follows:

$$\big[\mathbf{u}\big]_{\mathcal{B}} = \begin{bmatrix} u^1 \\ u^2 \\ \vdots \\ u^n \end{bmatrix} \qquad \big[\mathbf{u}\big]_{\tilde{\mathcal{B}}} = \begin{bmatrix} \tilde{u}^1 \\ \tilde{u}^2 \\ \vdots \\ \tilde{u}^n \end{bmatrix} \tag{23}$$

$$(\mathbf{M}_{\mathcal{B}\to\tilde{\mathcal{B}}})^{-1} \cdot \big[\mathbf{u}\big]_{\mathcal{B}} = \big[\mathbf{u}\big]_{\tilde{\mathcal{B}}} \tag{24}$$

I do not provide a proof for this, but it is easy to show that this is the case. This kind of behavior is called a contravariant transformation because components of $\mathbf{u}$ transform with the inverse of the change-of-basis matrix.

There is a vector space which is called the dual space of $\mathbf{V}$, denoted by $\mathbf{V}^*$.

$$\mathbf{V}^* := \{f : \mathbf{V} \to \mathbb{R} \mid f \text{ is linear}\} \tag{25}$$

In other words, $\mathbf{V}^*$ is defined as the space of linear functions from $\mathbf{V}$ to underlying field $\mathbb{R}$. Such functions are also called functionals. It turns out that $\mathbf{V}^*$ admits a canonical vector space structure. $\{\mathbf{V}^*, +_{\mathbf{v}^*}, \cdot_{\mathbb{R}}\}$

$$(f +_{\mathbf{v}^*} g)(\mathbf{u}) = f(\mathbf{u}) + g(\mathbf{u}) \tag{26}$$

$$(c \cdot_{\mathbb{R}} f)(\mathbf{u}) = c \cdot f(\mathbf{u}) \tag{27}$$

One can check that with defined vector addition and scalar multiplication, $\mathbf{V}^*$ is indeed a vector space.

Given a basis $\mathcal{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ for the original space $\mathbf{V}$, one can canonically the define dual basis $\mathcal{B}^* = \{\mathbf{v}^1, \ldots, \mathbf{v}^n\}$ for $\mathbf{V}^*$ where,

$$\mathbf{v^i} : \mathbf{V} \to \mathbb{R} \quad \text{ and } \mathbf{v^i}(\mathbf{v_j}) = \delta^i_j \tag{28}$$

δ^i_j is called a Kronecker-delta symbol and defined as follows:

$$\delta^i_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \tag{29}$$

We can represent α in terms of the dual basis of $\mathcal{B}^*$ as follows.

$$\alpha = \sum_{i=1}^{n} \alpha_i \mathbf{v}^i \tag{30}$$

or using the similar convention,

$$\big[\alpha\big]_{\mathcal{B}^*} = \begin{bmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_n \end{bmatrix} \tag{31}$$

I have a good reason for sticking with above convention. You might have noticed that when representing elements $\mathbf{u} \in \mathbf{V}$, I represent their components using upper index u^i and represent $\mathbf{u}$ with a column vector for a given basis $\mathcal{B}$, whereas I represent elements $\alpha \in \mathbf{V}^*$ by down index α_i and with a row vector.

Now check this out:

$$\begin{gathered} \alpha = (\alpha_1 \mathbf{v}^1 + \alpha_2 \mathbf{v}^2 + \cdots + \alpha_n \mathbf{v}^n) \\ \alpha(\mathbf{u}) = \alpha(u^1 \mathbf{v}_1 + u^2 \mathbf{v}_2 + \cdots + u^n \mathbf{v}_n) \\ \alpha(u^1 \mathbf{v}_1 + u^2 \mathbf{v}_2 + \cdots + u^n \mathbf{v}_n) =_{\text{linearity}} u^1 \alpha(\mathbf{v}_1) + u^2 \alpha(\mathbf{v}_2) + \cdots + u^n \alpha(\mathbf{v}_n) \\ u^1(\alpha_1 \mathbf{v}^1 + \alpha_2 \mathbf{v}^2 + \cdots + \alpha_n \mathbf{v}^n)(\mathbf{v}_1) + \cdots + u^n(\alpha_1 \mathbf{v}^1 + \alpha_2 \mathbf{v}^2 + \cdots + \alpha_n \mathbf{v}^n)(\mathbf{v}_n) \end{gathered}$$

Expanding last expression further, let us analyze the expression

$$u^i(\alpha_1 \mathbf{v}^1 + \alpha_2 \mathbf{v}^2 + \cdots + \alpha_n \mathbf{v}^n)(\mathbf{v_i}) = u^i(\alpha_1 \mathbf{v}^1(\mathbf{v_i}) + \cdots + \alpha_n \mathbf{v}^n(\mathbf{v_i})) \tag{32}$$

Using the definition $\mathbf{v}^i(\mathbf{v}_j) = \delta^j_i$, one can observe that the only surviving term from eq(32) is

$$u^i(\alpha_1 \mathbf{v}^1(\mathbf{v_i}) + \cdots + \alpha_n \mathbf{v}^n(\mathbf{v_i})) = u^i \alpha_i \tag{33}$$

Then using this knowledge, one can see that

$$\alpha(\mathbf{u}) = \big[\alpha\big]_{\mathcal{B}^*} \cdot \big[\mathbf{u}\big]_{\mathcal{B}} = \begin{bmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_n \end{bmatrix} \cdot \begin{bmatrix} u^1 \\ u^2 \\ \vdots \\ u^n \end{bmatrix} = \sum_{i=1}^{n} \alpha_i u^i \tag{34}$$

Now elements $\alpha \in \mathbf{V}^*$ are called covariant vectors, because when we change basis of $\mathbf{V}$ with matrix $\mathbf{M}_{\mathcal{B}\to\tilde{\mathcal{B}}}$ the components of the α changes with the same matrix. Thus, covariantly. Again, I do not provide a proof for that, but the general transformation is,

$$\left[\alpha\right]_{\tilde{\mathcal{B}}} = \left[\alpha\right]_{\mathcal{B}} \cdot \mathbf{M}_{\mathcal{B}\to\tilde{\mathcal{B}}} \tag{35}$$

Intuitively, all that formulation of dual spaces is to provide a machinery for us to construct functions that eats vectors and spits real numbers. We will define general tensors using vectors and dual vectors (covectors, or 1-forms). For example, an element dual vector space can be thought of linear level planes as shown in the figure (8), when it eats a vector, it returns the number of level planes pierced by vectors. It is a machine that eats a vector and gives a real number in a linear fashion. Dual vectors are also called rank-1 covariant tensors. Dot product of

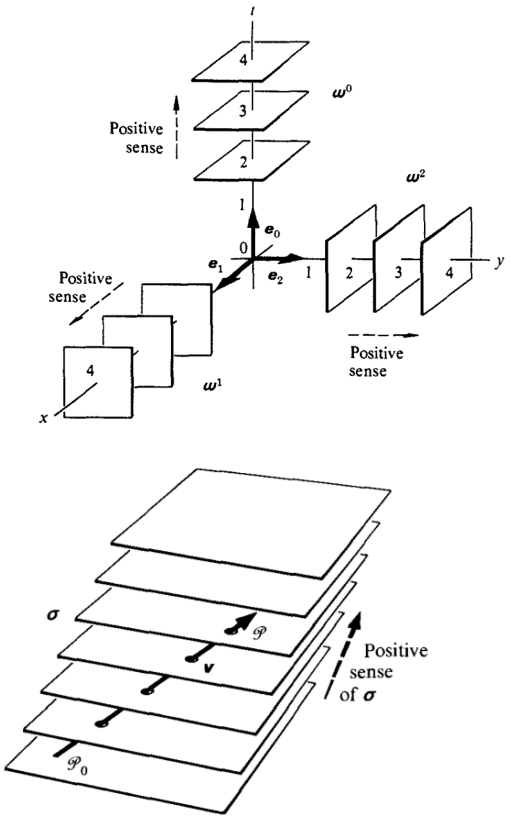

Figure 8: dual space as a level planes.

two vectors is rank-2 covariant symmetric, positive-definite tensor. We will use that terminology throughout this paper.

Back to the tensors, the general definition of the tensor is given as follows: An (m,n)-tensor on a finite-dimensional real vector space $\mathbf{V}$ is defined to be a multilinear map

$$\mathbf{T} : \underbrace{\mathbf{V} \times \mathbf{V} \dots \mathbf{V}}_{n} \times \underbrace{\mathbf{V}^* \times \mathbf{V}^* \times \cdots \times \mathbf{V}^*}_{m} \to \mathbb{R} \tag{36}$$

and the set of all such linear maps are represented by $\mathbb{T}_n^m(\mathbf{V})$

$$\mathbb{T}_n^m(\mathbf{V}) = \underbrace{\mathbf{V}^* \otimes \mathbf{V}^* \dots \mathbf{V}^*}_{n} \otimes \underbrace{\mathbf{V} \otimes \mathbf{V} \otimes \cdots \otimes \mathbf{V}}_{m} \tag{37}$$

Given a basis $\mathcal{B} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ for vector space every element $\mathbf{T}$ of (n,m)-tensor space can be represented as

$$T^{j_1, j_2, \dots, j_m}_{i_1, i_2, \dots, i_n} \mathbf{v}^{i_1} \otimes \mathbf{v}^{i_2} \otimes \cdots \otimes \mathbf{v}^{i_n} \otimes \mathbf{v}_{j_1} \otimes \mathbf{v}_{j_2} \otimes \cdots \otimes \mathbf{v}_{j_m} \tag{38}$$

There is summation over each index, but we follow the **Einstein summation convention**: if an index appears both as an upper (contravariant) and lower (covariant) index in a term, it is implicitly summed over, and we omit the summation symbol. In eq.(38), the tensor basis element

$$\mathbf{v}^{i_1} \otimes \cdots \otimes \mathbf{v}^{i_n} \otimes \mathbf{v}_{j_1} \otimes \cdots \otimes \mathbf{v}_{j_m}$$

is a multilinear map that eats n vectors and m covectors and returns a real number. Its value on basis vectors is defined analogously to a generalized Kronecker delta:

$$\left(\mathbf{v}^{i_1} \otimes \cdots \otimes \mathbf{v}^{i_n} \otimes \mathbf{v}_{j_1} \otimes \cdots \otimes \mathbf{v}_{j_m}\right)\left(\mathbf{v}_{k_1}, \dots, \mathbf{v}_{k_n}, \mathbf{v}^{l_1}, \dots, \mathbf{v}^{l_m}\right) = \delta^{i_1}_{k_1} \dots \delta^{i_n}_{k_n} \delta^{l_1}_{j_1} \dots \delta^{l_m}_{j_m}$$

That is, the value is 1 if and only if all upper indices match their corresponding lower indices, and 0 otherwise. This definition generalizes the identity δ^i_j to higher-rank tensors.

3.2 Tensor Fields, Differentiating Tensors, Metric Tensor and Covariant Derivative

Main incentive for introducing tensors is that they give us a mathematical framework in which we can formulate our theories in a clear manner. Some tensor calculus identities will be provided in this section without proof. We will use these identities throughout the construction of our algorithm. So let me cut short and start introducing them.

1. **For a vector field of $\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^n$ or, equivalently, a (1,0)-tensor field**, **the gradient** or **total derivative of $\mathbf{A}$** is $n \times n$ **Jacobian matrix**:

$$\mathbf{JA} = d\mathbf{A} = \nabla \mathbf{A} = \left(\frac{\partial A_i}{\partial x_j}\right)_{ij} \tag{39}$$

2. Tensors are geometric objects, meaning that they are coordinate invariant: you can choose any coordinate system to represent your tensor field, when tensor field at a particular point acts on vectors and covectors it should yield the same value regardless of its representation. For instance, vectors are geometric objects: One can choose different basis to represent the same vector, but since vectors are geometric objects they have a unique transformation law under the change of basis, given in eq(23) and eq(35) for vectors and covectors respectively. What are we trying to achieve in this section is to define a notion of derivative for tensorial objects that is coordinate invariant. We want derivative to mean the same thing regardless of choice of coordinate system. To make this statement rigorous I need to introduce Smooth

Manifolds, and thus topology, but it is impossible for me to introduce these concepts in an introductory proposal. I believe, that, even without understanding of the formal structure of manifolds, these notions can be understood intuitively.

Tensors are objects that transform like a tensor. Tensor transformation laws can be deduced from their structure. (r,s)-tensor transforms under coordinate transformation as follows:

Let $\mathbf{T} : \mathbb{R}^n \to \mathbb{T}^r_s(\mathbb{R}^n)$ be an (r,s)-tensor field.

$$\left(T^{j_1,\ldots,j_r}_{i_1,\ldots,i_s}\right)_y = \frac{\partial y^{j_1}}{\partial x^{k_1}} \frac{\partial y^{j_2}}{\partial x^{k_2}} \cdots \frac{\partial y^{j_r}}{\partial x^{k_r}} \frac{\partial x^{m_1}}{\partial x^{i_1}} \frac{\partial x^{m_2}}{\partial x^{i_2}} \cdots \frac{\partial x^{m_s}}{\partial x^{i_s}} \left(T^{k_1,\ldots k_r}_{m_1,\ldots,m_s}\right)_x \tag{40}$$

Where $\left(T^{j_1,\ldots,j_r}_{i_1,\ldots,i_s}\right)_y$ represents components of the tensor $\mathbf{T}$ in y coordinate. This transformation rule is a mix of covector and vector component transformations as given in eq(23) and eq(35). Since we are talking about tensor fields, you can imagine at each point $p \in \mathbb{R}^n$ there lives the tangent tensor space $T_p\mathbb{T}^r_s(\mathbb{R}^n)$, and at p, general coordinate transformation $y : \mathbb{R}^n \to \mathbb{R}^n$ causes a change of basis in $T_p\mathbb{T}^r_s(\mathbb{R}^n)$ is given by $\frac{\partial y}{\partial x}$ evaluated at p. So at each point $p \in \mathbb{R}^n$, the change of basis matrix is the Jacobian $\mathbf{J}_p y$ of y at p. I also would like to add as a reminder that $(\mathbf{J}_p y)^{-1} = \frac{\partial x}{\partial y}$.[§]

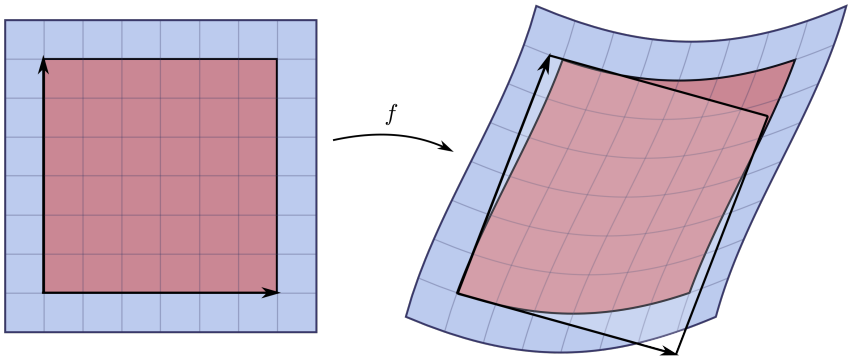

Figure 9: Jacobian of a Function as its best Linear Approximation

3. **Metric tensor** is a precious tool that imparts the notion of geometry, distances and angles to an abstract space. Imagine the real plane, choose two points, there is a distance function defined on that space that tells you how far apart these points are. However, metric and metric tensor are different things. Metric tensor will eventually give us a metric for overall space.

   Imagine a curved surface, surface of a sphere for instance, imagine we cut the surface into infinitesimal planes, and imagine at each of these planes we define a notion of distance between arbitrary two points. Each of these infinitesimal planes are what we call tangent vector space of the surface. We choose a symmetric, positive-definite (0,2)-tensor for each tangent vector space, -we define a tensor field over this surface. This (0,2)-tensor field will encode information of distance and angles, basically we attach a unique geometry to each tangent vector space. Unbelievable as it might sound, the information stored in this metric

[§] https://www.youtube.com/watch?v=XtpVVcKXfnA&list=PLdgVBOaXkb9D6zw47gsrtE5XqLeRPh27_&index=6

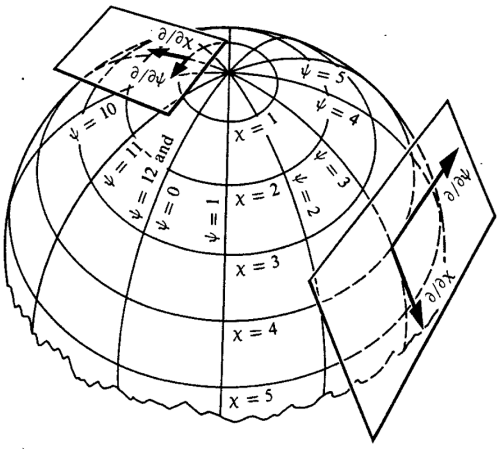

Figure 10: Surfaces as Augmentation of Tangent Vector Spaces

tensor is so great that we can reconstruct the surface from the metric tensor itself.[¶]

$$\mathbf{G} : \mathcal{M} \to \mathbb{T}_2^0(T\mathcal{M}) \tag{41}$$

$$\mathbf{G}_p = \big(g_{ij}(p)\big)_x \, d_p x^i \otimes d_p x^j \tag{42}$$

where $\mathcal{M} \subset \mathbb{R}^n$ is some embedded surface, given two vectors $\mathbf{X}, \mathbf{Y} \in T_p\mathcal{M}$

$$\mathbf{G}_p(\mathbf{X}, \mathbf{Y}) = \big(g_{ij}(p)\big)_x \, d_p x^i \otimes d_p x^j(\mathbf{X}, \mathbf{Y}) = \big(g_{ij}(p)\big)_x \, X^i Y^j \tag{43}$$

Metric tensor being symmetric is equivalent to say that for given arbitrary two vectors $\mathbf{X}, \mathbf{Y} \in T_p\mathcal{M}$, $\mathbf{G}_p(\mathbf{X}, \mathbf{Y}) = \mathbf{G}_p(\mathbf{Y}, \mathbf{X})$ for all $p \in \mathcal{M}$

4. **Covariant derivative**[‖] of a tensor field is one of the three possible, structurally consistent ways of defining derivatives of tensor fields, two other being **exterior derivative** and **Lie Derivative**. To define the covariant derivative we need a metric tensor over our $\mathbb{R}^n$, that is, an inner product defined over every **tangent vector space** on our space. We will use covariant derivative to solve Partial Stochastic Differential Equations in our parameter space and to set up advanced optimization methods, and only by using these differential geometric - tensorial tools we can achieve that. Covariant Derivative of a vector field $\mathbf{X}$ with respect to a vector field $\mathbf{V}$ is given by Covariant derivative operator $\nabla : \mathcal{T}\mathcal{M} \times \mathcal{T}M \to \mathbb{T}_1^1(\mathcal{T}\mathcal{M})$, explicitly:

$$\nabla(\mathbf{X}, \mathbf{V})_j^i = (\nabla_{\mathbf{v}}\mathbf{X})_j^i = \frac{\partial X^i}{\partial x^j} + \Gamma_{jk}^i X^j X^k \tag{44}$$

More generally, it can be defined for arbitrary (n,m) tensor field by the relation

$$\nabla_{\mathbf{v}}(\phi \otimes \psi) = \nabla_{\mathbf{v}}\phi \otimes \psi + \nabla_{\mathbf{v}}\psi \otimes \phi \tag{45}$$

where ϕ and ψ are tensors whose ranks sums up to (n,m), the result is (n,m+1) tensor field because when you try to transform this object, you observe that it does indeed transform

[¶] https://www.youtube.com/watch?v=sV58Fy2s6ac
[‖] https://www.youtube.com/watch?v=BHKd6-IJgVI

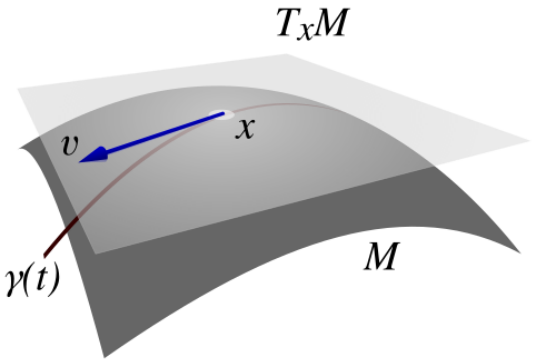

Figure 11: Tangent Vector space at $x \in \mathcal{M}$, denoted by $T_x\mathcal{M}$

like an (n,m+1) tensor. The mysterious term Γ^i_{jk} appearing in the right hand side of eq(44) is called **Christoffel Symbols** and they are peculiar objects, also called **connection coefficients**, they allow you to generalize notion of derivative to curved spaces, and this term connects**. Connection coefficients are not tensorial objects, but they are rather peculiar differential geometric objects that transform under the change of coordinates $x \to \tilde{x}$ in the following manner,

$$\tilde{\Gamma}^m_{pq} = \frac{\partial x^i}{\partial \tilde{x}^p}\frac{\partial x^j}{\partial \tilde{x}^q}\frac{\partial \tilde{x}^m}{\partial x^k}\Gamma^k_{ij} + \frac{\partial^2 x^s}{\partial \tilde{x}^p \partial \tilde{x}^q}\frac{\partial \tilde{x}^m}{\partial x^s} \tag{46}$$

The intuition behind this definition can be justified by analyzing the structure of naive tensorial derivative and why they failed to be in a tensorial form, to resolve that issue it turns out that the most natural structure to add is connection coefficients.

If we are given a metric **G**,

$$\mathbf{G} : \mathcal{M} \to \mathbb{T}^0_2(T\mathcal{M}) \tag{47}$$

Then there is a unique connection induced by the metric up to the some regularity conditions, this connection coefficient has a special name: **Levi-Civita Connection**. In an arbitrary x coordinates, the coefficient of the connection are given by the following expression:

$$\Gamma^k_{ij} = \frac{1}{2}g^{kl}\left(\frac{\partial g_{jl}}{\partial x^i} + \frac{\partial g_{il}}{\partial x^j} - \frac{\partial g_{ij}}{\partial x^l}\right) \tag{48}$$

Where g^{kl} is inverse of the metric components in the given x-coordinate system. The whole motivation is that, when working with financial data, we regard them as multidimensional surfaces that has a unique metric, the distance between them that we will define as we see fit, and we want to do calculus on that metric and thus solve Stochastic partial differential equations on that space in order to perform financial optimiziation.

**https://www.youtube.com/watch?v=TvFvL_sMg4g&t=616s

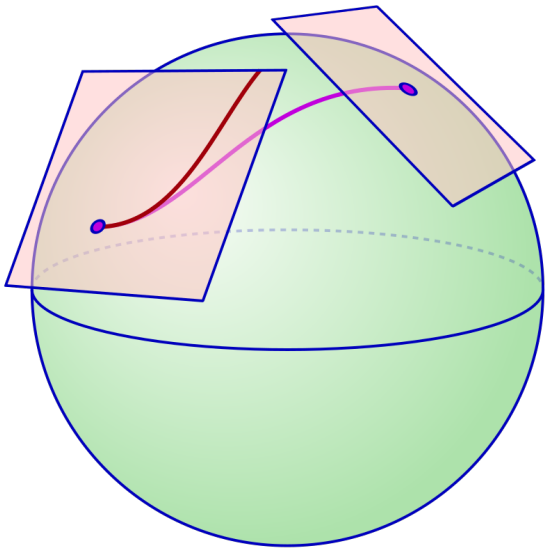

Figure 12: Covariant derivative as an object that connects two nearby Tangent Vector Spaces on surface $\mathcal{M}$

3.3 Matrix Calculus as a Special Case of Tensor Calculus

3.4 Backpropagation in its Most General Form

Let us dive into the backpropagation and have us in our minds the tensorial formalism, I will work in the most general fully connected feed-forward neural network architecture possible, and therefore I will propose few differences to what I introduced in the first section regarding the structure of neural nets. First of all, the activation function for each layer doesn't have to be in the same form, we can use different activation functions for each layer. The activation function for the layer L will be denoted by ϕ^L.

$$\mathcal{L}(\mathcal{M}) = \sum_i \mathcal{D}(\mathcal{F}_{\mathcal{M}}(\mathbf{X^i}), \mathbf{Y^i}) \tag{49}$$

Where $\mathcal{D} : \mathbb{R}^{N_1} \times \mathbb{R}^{N_k} \to \mathbb{R}$ is an arbitrary smooth function. We want to calculate the gradient of the loss function $\mathcal{L}_{\mathcal{M}}$ symbolically $\nabla_{\mathcal{M}}\mathcal{L}$, the subindex $\mathcal{M}$ means that we take this gradient with respect to trainable model parameters (weights and biases). The other difference is in the cost function: so I will decompose these two into their own separate form. Let us start with the gradient of the loss function with respect to weights (that is, in which direction should I change the weights in order to move in the direction where the error decreases the most)

$$\frac{d\mathcal{L}}{d\mathbf{X}} = \frac{d\mathcal{L}}{d\sigma^{\mathbf{k}}}\frac{d\sigma^{\mathbf{k}}}{d\mathbf{z^k}}\frac{d\mathbf{z^k}}{d\sigma^{\mathbf{k-1}}}\cdots\frac{d\sigma^{\mathbf{2}}}{d\mathbf{z^2}}\frac{d\mathbf{z^2}}{d\mathbf{X}} \tag{50}$$

This follows by the chain rule. Now there is a slight change in the notation, the above derivatives represent total derivatives. But since these are multi-dimensional objects (we are representing vector quantity with respect to matrix quantity for instance when we will differentiate $\frac{d\mathbf{z^k}}{d\mathbf{W^{k-1}}}$) we are actually dealing with tensorial qauntities.

$$\mathbf{T} = \frac{d\mathbf{z^k}}{d\mathbf{W^{k-1}}} \tag{51}$$

where

$$T_{ij}^{s} = \left(\frac{d\mathbf{z^k}}{d\mathbf{W^{k-1}}}\right)_{ij}^{s} = \frac{\partial z_s^k}{\partial W_{ij}^{k-1}} \tag{52}$$

Explicitly

$$\frac{\partial z_s^k}{\partial W_{ij}^{k-1}} = \frac{\partial}{\partial W_{ij}^{k-1}}\left(\sum_n W_{sn}^{k-1} a_n^{k-1} + b_s^k\right) = \left(\sum_n W_{sn}^{k-1} a_n^{k-1} \delta_{is}\delta_{nj}\right) \tag{53}$$

Where δ_{is} and δ_{nj} are Kronecker-Delta symbols as usual. Let us define another quantity which will contain in itself the characteristic properties of neural network itself, this quantity and its iterative definition will be of at most importance, and tell us all the facts regarding the behavior of the gradient descent on that model with given set of activation functions and particular loss function. The analysis of such a nonlinear recursive relation will require functional analysis and in particular spectral theorem.

$$\xi^{\mathbf{k}} = \frac{\mathbf{d}\mathcal{L}}{\mathbf{d}\sigma^{\mathbf{k}}}\frac{\mathbf{d}\sigma^{\mathbf{k}}}{\mathbf{dz^k}} \tag{54}$$

$$\xi^{k-1} = \xi^k \cdot \mathbf{W}^{k-1} \cdot \phi^k(\mathbf{z^{k-1}}) \tag{55}$$

That is all, we have the backpropagation.

4 News Reading and Sentiment Analysis

5 Probability and Stochastic Processes

Probabilities do play an important role both in modeling of financial systems, statistics and naturally in neural net algorithms. It is quite important to know how to define the concept of probability in rigorous mathematical settings in order for us to manipulate them further and thus be able to make meaningful statements regarding the overall system we are building. We need to integrate probabilities in our model, for us to do that we need to make few mathematical assumptions about these relations. For instance we will model economic markets as Brownian motions, and we define what Brownian motion is in terms of random variables. The game is simple actually, we need concepts, and we need mathematically, so we define and refine, define and refine, and make critical assumptions and prove them. For us to even begin manipulating these object we first must define them rigorously

5.1 Defining Fundamental Concepts

A probability space is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a sample space, $\mathcal{F} \subset \mathcal{P}(\Omega)$, where $\mathcal{P}(\Omega)$ is the power set of Ω, is the event space and $\mathbb{P} : \mathcal{F} \to \mathbb{R}$ is a probability measure. For instance, one can imagine the sample space as being the set of all possible configurations triple coin tosses. That is $\Omega = \{111, 110, 101, 011, 100, 010, 001, 000\}$ and event space can be considered simply as the power set of Ω. In that case if we ask the question, what is the probability that when I toss a coin three times in a row, at least two consecutive coins has the same side. That would require us to select all possible cases where such is the case, call that an event $\mathcal{A} \in \mathcal{F}$ where $\mathcal{A} = \{111, 110, 011, 100, 001, 000\}$ and we define the probability of $\mathcal{A}$ as $\mathbb{P}(\mathcal{A})$, since our space is discreet, we can define the probability of an event $\mathcal{A} \in \mathcal{F}$ as $\mathbb{P}(\mathcal{A}) = \frac{|\mathcal{A}|}{|\Omega|}$ where $|\mathcal{A}|$ = number of elements in A. This gives us a good intuition for the definition of probability

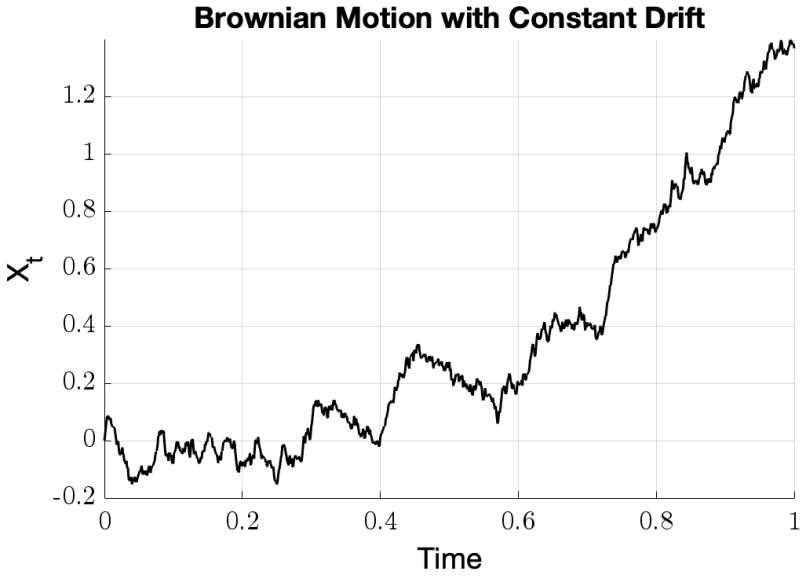

Figure 13: Simple Brownian Motion with constant upward drift coefficient

space, but we must axiomatize such notions to more general settings, for instance how can we quantify the probability measure if our sample space contains infinitely many elements?

Probability measure $\mathbb{P}$ must satisfy three conditions

$$\mathbb{P}(\Omega) = 1 \tag{56}$$

$$\mathbb{P}\left(\bigcup_i \mathcal{A}_i\right) = \sum_i \mathbb{P}(\mathcal{A}_i) \quad \text{for } \mathcal{A}_i \bigcap \mathcal{A}_j = \emptyset \text{ for all } i \neq j \text{ where } \mathcal{A}_i \in \mathcal{F} \tag{57}$$

Let's make sense of these two conditions and understand why they are defined the way they are. Eq(51) says that if an event in consideration is all there is that is possible, event must occur. If two events have nothing in common, the probability that both is occurring is sum of each event being occurring distinctively. From the definitions we can also deduce further properties of probability measure.

$$\mathbb{P}(\mathcal{A}) + P(\Omega \backslash \mathcal{A}) = 1 \tag{58}$$

The probability that event occuring or not occuring is always 1.

$$\mathbb{P}(\mathcal{A} \bigcup \mathcal{B}) = \mathbb{P}(\mathcal{A}) + \mathbb{P}(\mathcal{B}) - \mathbb{P}(\mathcal{A} \bigcap \mathcal{B}) \tag{59}$$

$$\mathbb{P}(\emptyset) = 0 \tag{60}$$

Now that we are given axioms of probability measure, we also must axiomatize what is allowed to be in the event space $\mathcal{F}$

$$\text{if } \mathcal{A} \in \mathcal{F} \text{ so is its complement } \mathcal{A}^c = \Omega \backslash \mathcal{A} \tag{61}$$

$$\text{if countably many events } \{\mathcal{A}_1, \mathcal{A}_2, \dots\} \in \mathcal{F} \text{ so is } \bigcup_{i\in\mathbb{N}} \mathcal{A}_i \tag{62}$$

Intuitively, it says that if something that can happen is an event, something that is not happening is also an event; and, if there are countable number of events, so is all of them happening is an event. Using a little set theoretic relations -boolean algebra-, one can also deduce the fact that

$$\text{if countably many sequence of events } \{\mathcal{A}_1, \mathcal{A}_2, \dots\} \in \mathcal{F} \text{ so is } \bigcap_{i\in\mathbb{N}} \mathcal{A}_i \tag{63}$$

A **random variable** $\mathbf{X}$ is a function $\mathbf{X} : \Omega \to \mathbb{R}$ such that the set $\{\mathbf{X} = c\} \in \mathcal{F}$; where $\{\mathbf{X} = c\} \subset \Omega$ is defined as the set $\{\omega \in \Omega \mid \mathbf{X}(\omega) = c\}$. Basically we want to measure the set of points ω in sample space that has the property $\mathbf{X}(\omega) = c$. Random variables are ways to partition and quantify the sample space. Consider the example of coin toss above, one can imagine a function $\mathbf{X} : \{111, 110, 101, 011, 100, 010, 001, 000\} \to \mathbb{R}$, where $\mathbf{X}$ is defined as follows:

$$\mathbf{X}(\omega) = \begin{cases} 1 & \text{if } \omega \text{ has at least two consecutive 0's or 1's} \\ 0 & \text{otherwise} \end{cases} \tag{64}$$

Looking at the sets $\{\mathbf{X} = 1\} = \mathcal{A}$ and $\{\mathbf{X} = 0\} = \mathcal{A}^c$, we immediately see that they are in the event space and that, $\mathbb{P}(\{\mathbf{X} = 1\}) = \frac{6}{8}$ and that $\mathbb{P}(\{\mathbf{X} = 0\}) = \frac{2}{8}$ with the probability measure defined as in the example above. Here, one can imagine $\mathbf{X}$ as a way of labeling and classifying the events.

Now, it turns out that, in order for us to do calculus on probabilities, we need continuous sample spaces, and probability distributions. Here, I'll introduce the general definition of the **Cumulative Distribution Function(CDF) of a given random variable** $\mathbf{X} : \Omega \to \mathbb{R}$, which we will denote by $F_{\mathbf{X}} : \mathbb{R} \to \mathbb{R}$, where it is defined as follows:

$$F_{\mathbf{X}}(x) = \mathbb{P}(\{\mathbf{X} \leq x)\} \tag{65}$$

If $F_{\mathbf{X}}$ is an **absolutely continuous function**, it admits a unique **Probability DENSITY Function (PDF)**

$$P_{\mathbf{X}} = \frac{dF_{\mathbf{X}}}{dx} \tag{66}$$

Taylor expanding the CDF of X, we obtain

$$F_{\mathbf{X}}(x + \Delta x) = F_{\mathbf{X}}(x) + \frac{dF_{\mathbf{X}}}{dx}|_x \Delta x \qquad \text{as } \Delta x \to 0 \tag{67}$$

Manipulating this expression yields

$$F_{\mathbf{X}}(x + \Delta x) - F_{\mathbf{X}}(x) = \frac{dF_{\mathbf{X}}}{dx}|_x \Delta x \qquad \text{as } \Delta x \to 0 \tag{68}$$

Now, since

$$F_{\mathbf{X}}(x + \Delta x) = \mathbb{P}\left(\{X \leq x + \Delta x\}\right) \tag{69}$$

and that

$$F_{\mathbf{X}}(x) = \mathbb{P}\left(\{X \leq x\}\right) \tag{70}$$

Thus,

$$F_{\mathbf{X}}(x + \Delta x) - F_{\mathbf{X}}(x) = P_{\mathbf{X}}(x)\Delta x = \mathbb{P}(\{x \leq X \leq x + \Delta x\}) \tag{71}$$

Above expression makes it clear as to what probability density means: it is a function that essentially gives the probability of $x \leq \mathbf{X} \leq x + \Delta x$ when multiplied by Δx, where Δx is an infinitesimal quantity. More rigorously, it is the best linear approximation of CDF around x. Employing fundamental theorem of calculus one observes that integrating the density function in the interval (a, b) gives the probability of random variable $\mathbf{X}$ having the value in (a, b).

$$\int_a^b P_{\mathbf{X}}(x)dx = F(b) - F(a) = \mathbb{P}(\{a \leq X \leq b\}) \tag{72}$$

In the discrete case, one has the Probability mass function, denoted by

$$p_{\mathbf{x}}(a) = \mathbb{P}(\{X = a\}) \tag{73}$$

Let us consider an example of the continuous case: Imagine that you are trying to understand the distribution of human height, for that you choose 1000 random people from all around the world. You partition heights with $h_0 = (0cm, 2cm], h_1 = (2cm, 4cm], \ldots, h_n = (2n, 2n + 2]$ classes, you measure heights of each of 1000 people, and for each class you count how many people's height belong to this class. You will approximately obtain the following histogram: If you let number of

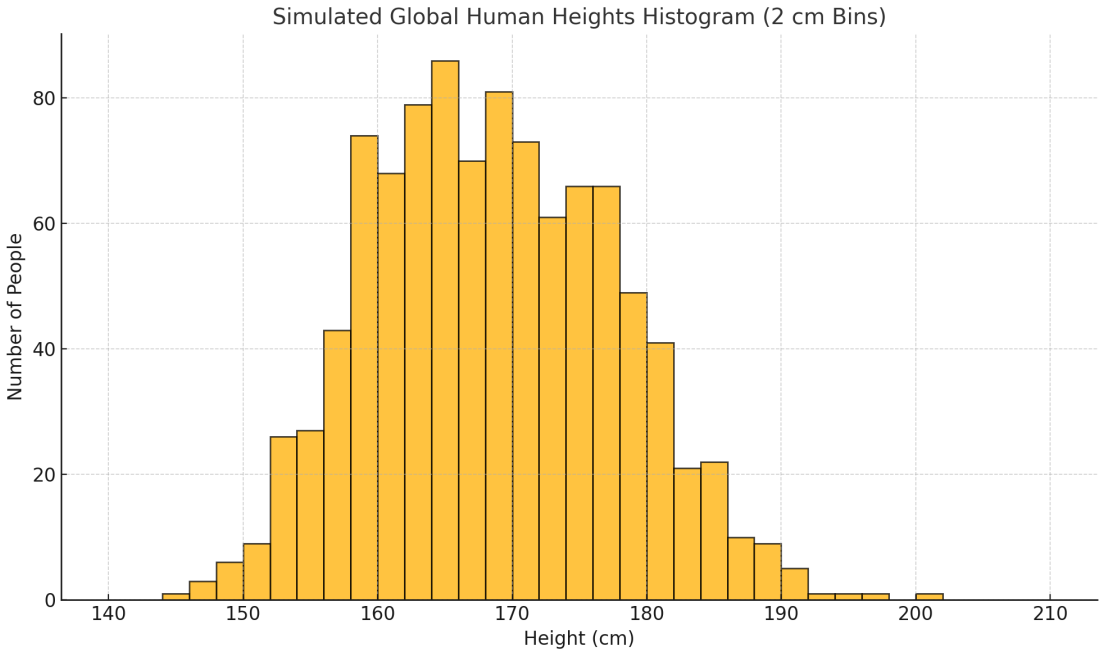

Figure 14: Caption

people you sampled go to infinity, and if you let the width of the bins to 0, you would get what is known as **normal distribution**, also known as **Gaussian Distribution**. It is the probability density $P_{\mathbf{X}}$, where our sample space $\Omega = \mathbb{R}^+$ and the random variable $\mathbf{X} : \Omega \to \mathbb{R}$ that takes an height and gives the "continuous" number of people having that height. Probability measure induced by the random variable $\mathbf{X}$, $\mathbb{P}_{\mathbf{X}}$, is given by,

$$\mathbb{P}(\mathbf{X} \in \mathcal{A}) = \int_{\mathcal{A}} P_{\mathbf{X}}(x)dx \tag{74}$$

and that

$$P_{\mathbf{X}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(\frac{x-\mu}{\sigma}\right)^2} \tag{75}$$

Now, as always, we generalize the notion of probability density, and consider all continuous, bounded, non-negative function that integrates to 1 over $\mathbb{R}$ to be a density of some probability measure. This is such a beautiful classification as it will allow us to consider exotic probability densities to model random fluctuations of the financial markets. There are billions of such

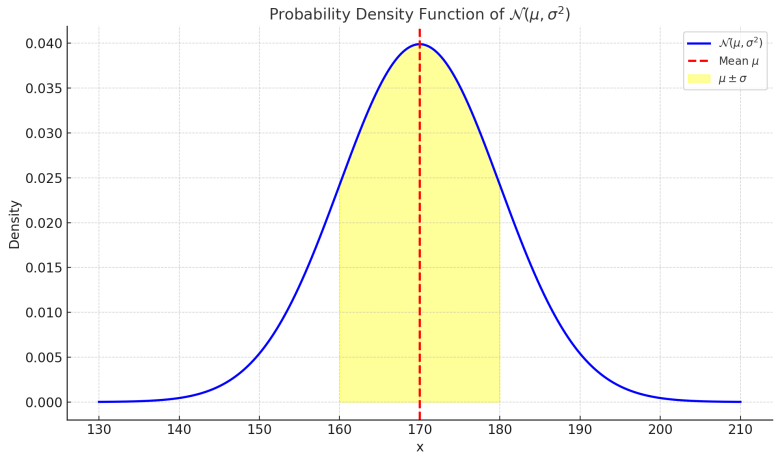

Figure 15: Normal distribution with mean $\mu = 170$ and standard deviation σ

functions each of which describes a different process. The study of such functions is a branch of statistics. For instance, given a finite set of samples, is it possible that one can predict the probability density of such samples as a function of several parameters (known as observables)? Kernel Density Approximation is one algorithm to construct such functions. Asking the reverse of that question, can one correctly sample from a given probability distribution? Monte Carlo Markov Chain algorithm is one of the most important algorithms from 20th century, it is an efficient algorithm that uses Markov chains to sample points from given probability distribution.

5.2 Expectation, Variance, Multivariable Distributions, Conditional Probabilities, Independence and Bayes' Rule and Characteristic Function

Tools that will be introduced in this section are powerful when we want to analyze processes, by defining these notions, we use these terms later to define what a martingale is, or what a markov process is, so it is essential for us to get a strong grasp of these notions, and we need to study their properties. Let us start with the definition of an expectation. Expectation $\mathbb{E}[\mathbf{X}]$ of a random variable $\mathbf{X} : \Omega \to \mathbb{R}$ is defined as:

$$\mathbb{E}[\mathbf{X}] = \int_{\mathbb{R}} x dF_{\mathbf{X}} \tag{76}$$

This integration is in the Lebesgue sense, and is defined similar to Riemann-Stieltjes integration. Defining Lebesgue integral is an easy job, let us first define what a simple function is:

$$S : \mathcal{F} \to \mathbb{R} \text{ such that } Im(S) = \{s_1, \ldots, s_n\} \text{ where } \{S = s_i\} \in \mathcal{F} \text{ for } i = 1, 2, \ldots, n \tag{77}$$

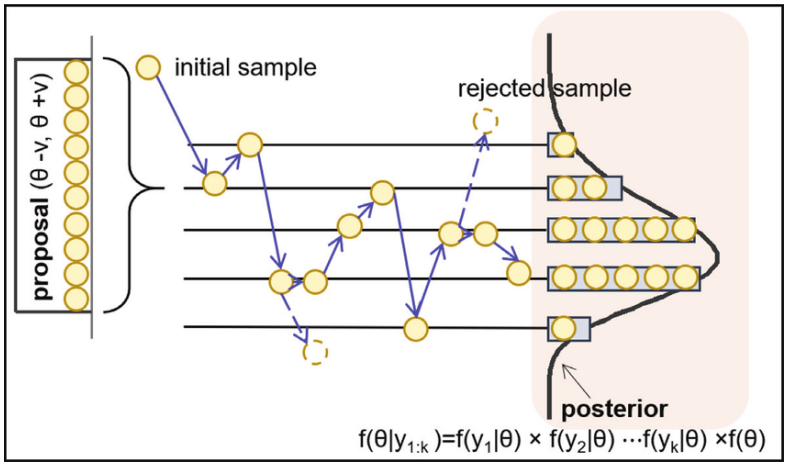

Figure 16: MCMC algorithm

Such functions are basically piecewise constant functions, and there are finite number of values in their codomain, and the set $\{S = s_i\}$ contained in the event space, which makes them also a random variable. Let us first define the Lebesgue integral of these simple functions:

$$\int S d\mathbb{P} = \sum_{i=1}^{n} s_i \cdot \mathbb{P}(\{S = s_i\}) \tag{78}$$

Now, similar to Riemann approach defining integration -first partitioning the real domain into small intervals and summing the value of functions for some point in that interval- we will partition the sample space Ω by simple functions that are approximating the non-simple function $f : \Omega \to \mathbb{R}$, and take the probability measure of that partition, and let that simple function approximation to go arbitrary precision. More formally

$$\int_{\mathbb{R}} f d\mathbb{P} = \sup_{S} \{ \int S d\mathbb{P} \quad | S \leq f \} \tag{79}$$

Now, we can basically perform this integration for any valid probability measure $\mathbb{P} : \mathcal{F} \to \mathbb{R}$, you can check that CDF induced by a random variable $\mathbf{X}$, $F_{\mathbf{X}} : \mathbb{R} \to \mathbb{R}$, is actually a valid probability measure(it satisfies the axioms of being probability measure). (Side Note: we actually defined the probability measure as a function from event space $\mathcal{F}$ to $\mathbb{R}$ but $F_{\mathbf{X}}$ has $\mathbb{R}$ as its domain. To resolve this seemingly paradoxical situation, we may change our above definition of random variable induced probability measure as a function $\tilde{F}_{\mathbf{X}} : \mathcal{F} \to \mathbb{R}$ where, for an event $\mathcal{A} \in \mathcal{F}$, $\tilde{F}_{\mathbf{X}}(\mathcal{A}) = \mathbb{P}(\{\mathbf{X} \subset U_{\mathcal{A}}\})$ where $\mathcal{A}$ is equal to the set $\{\mathbf{X} \subset U\}$ where $U_{\mathcal{A}} \subset \mathbb{R}$ is a closed interval, open interval or combination of them.)

Moreover, if CDF $F_{\mathbf{X}}$ is absolutely continuous, it is equivalent to ordinary Riemann-Stieltjes integral defined by:

$$\mathbb{E}[\mathbf{X}] = \int_{\mathbb{R}} x P_{\mathbf{X}} dx \tag{80}$$

This is the form that will be considered in our implementation, we will work with densities and integrals we know how to work with. Now, all this mathematical masturbation was necessary

for us to generalize the concept of 'average' or 'mean' to probabilities of wildest form, (we will be dealing with these guys, eventually, when we try to beat the market).

To have an intuition about it, try to think about what would happen in the discrete case, consider the coin toss as an example with the $p_{\mathbf{x}} : \mathcal{F} \to \{0, 1\}$ we defined above $p_{\mathbf{x}}(c) = \mathbb{P}(\{\mathbf{X} = c)\}$, and try to figure out the expectation of the random variable that is defined as above (consecutive coin tosses = 1, otherwise =0), discrete case will simplify to,

$$\sum_{c \in \{0,1\}} x \cdot p_{\mathbf{x}} = 0 + 1(6/8) = 6/8 \tag{81}$$

Basically we attach each possible situation a weight given by the induced probability measure through the random variable $\mathbf{X}$ and sum all possible values, the most influential values are the ones with most weight (most probable ones), and usually the expected value of a random variable is what is most likely to happen. Same goes for the continuous case, intuition is the same, calculus is just the tool we use to define such a concept in mathematically consistent manner. For instance if we consider a random variable whose induced CDF admits a gaussian density, it would have a mean μ. Formally,

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{\mathbb{R}} x e^{\left(\frac{x-\mu}{\sigma}\right)^2} dx = \mu \tag{82}$$

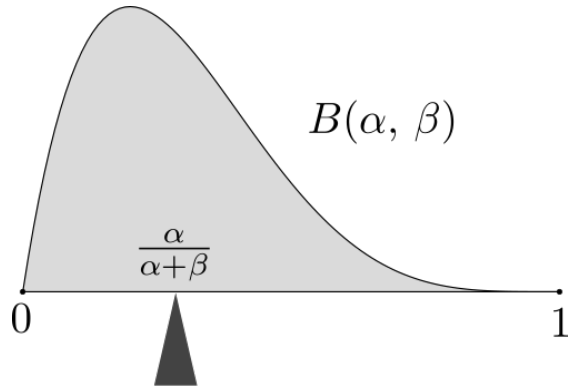

Figure 17: Expected value as a center of mass of the probability density

Now, we want to capture the notion of how spread the probability distribution is, to that, let us define it precisely and then intuit about it. For a given random variable $\mathbf{X} : \Omega \to \mathbb{R}$, the **variance** $\mathbf{V}[X]$ is defined as,

$$\mathbf{V}[X] = \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])^2 \tag{83}$$

To understand structure of this equation, remember that $\mathbb{E}[\mathbf{X}]$ is a definite real number, it is constant, and we define $\mathbf{X} - \mathbb{E}[\mathbf{X}] : \Omega \to \mathbb{R}$ and it is defined by $(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\omega) = \mathbf{X}(\omega) - \mathbb{E}[\mathbf{X}]$, other operations are defined similarly.

Attention! Mathematical Masturbation Ahead: Let us talk about independence of random variables. In order to do that, we need to introduce the concept of $\sigma - algebra$. Given any set Ω, $\sigma - algebra$ $\Sigma \subseteq \mathcal{P}(\Omega)$ is a family of subsets of Ω, such that,

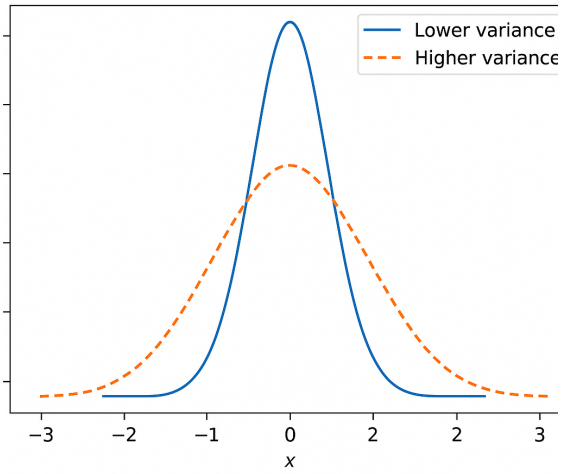

Figure 18: Variances of Two Different Normal Distributions

1. $\Omega \in \Sigma$
2. if $\mathcal{A} \in \Sigma \implies \mathcal{A}^c \in \pm$
3. $$\text{if countably many events } \{\mathcal{A}_1, \mathcal{A}_2, \dots\} \in \Sigma \text{ so is } \bigcup_{i \in \mathbb{N}} \mathcal{A}_i \tag{84}$$

   Event space is just a $\sigma - algebra$ we assigned to the sample space. The tuple (Ω, Σ) is called a measurable space, and triple (Ω, Σ, μ) is called a measure space, where μ is a general measure. Probability space is special measure space, with the condition that $\mu(\Omega) = 1$.

We also need to introduce the standard $\sigma - algebra$ of $\mathbb{R}$, also known as Borel $\sigma - algebra$. It is a special $\sigma - algebra$ on $\mathbb{R}$. One can define a standard metric-topology on $\mathbb{R}$, denoted by $\mathcal{O}_{\mathbb{R}} \subset \mathcal{P}(\mathbb{R})$(bkz. metric topology on $\mathbb{R}$), using that structure, one can define Borel $\sigma - algebra$ as the smallest possible $\sigma - algebra$ that contains $\mathcal{O}_{\mathbb{R}}$; it is denoted by $\sigma(\mathcal{O}_{\mathbb{R}})$. Having that notion in mind, we generalize a random variable $\mathbf{X} : \Omega \to \mathbb{R}$ by relaxing the former condition to

$$\mathbf{X}^{-1}(A) \in \mathcal{F} \text{ where } A \in \sigma(\mathcal{O}_{\mathbb{R}}) \tag{85}$$

In words: The set of points in Ω that is mapped to a Borel measurable set $A \in \sigma(\mathcal{O}_{\mathbb{R}})$ is $\mathcal{F}$ measurable.

Now we are ready to define independence of two random variables. Given a random variable $\mathbf{X} : \Omega \to \mathbb{R}$, $\sigma - algebra$ generated by the random variable $\mathbf{X}$ is the smallest $\sigma - algebra$ that contains the family of subsets of Ω given by,

$$\sigma(\mathbf{X}) = \sigma(\{\mathbf{X}^{-1}(A) \in \mathcal{F} \quad | \quad \text{for all } A \in \sigma(\mathcal{O}_{\mathbb{R}}\})) \tag{86}$$

We say that $\sigma - algebras$ $\mathcal{B}$ and $\mathcal{A}$ are independent if

$$\mathbb{P}(A \cap B) = \mathbb{P}(A) \cdot \mathbb{P}(B) \tag{87}$$

for any $A \in \mathcal{A}$ and $B \in \mathcal{B}$, and we say that two random variables are independent if sigma algebras induced from them are independent. Can we encode that information into probability densities somehow? Lets try that, we know that for a given random variable $\mathbf{X}$, $F_{\mathbf{X}} : \mathbb{R} \to \mathbb{R}$ and $F_{\mathbf{Y}} : \mathbb{R} \to \mathbb{R}$ for a given random variable $\mathbf{Y}$. Assume that $\mathbf{X}$ and $\mathbf{Y}$ are independent random variables. Then,

$$F_{\mathbf{X}}(x) = \mathbb{P}(\{\mathbf{X} = x\}) \tag{88}$$

$$F_{\mathbf{Y}}(y) = \mathbb{P}(\{\mathbf{Y} = y\}) \tag{89}$$

$$F_{\mathbf{X},\mathbf{Y}}(x, y) = \mathbb{P}(\{\mathbf{X} = x\} \cap \{\mathbf{Y} = y\}) = F_{\mathbf{X}}(x)F_{\mathbf{Y}}(y) \tag{90}$$

$$\int \frac{\partial F_{\mathbf{X},\mathbf{Y}}(x, y)}{\partial x} dy = P_{\mathbf{X}} \tag{91}$$

[To be documented Later]

5.3 Calculus of Probabilities

5.4 Markov Processes

5.5 Brownian Motion and Diffusion Process

5.5.1 Ito's Lemma

5.6 Fokker-Plank Equation

5.7 Black-Scholes Equation and Few words on Financial Models

5.8 Analyzing Financial Markets and Statistical Mechanics

6 Neural Networks Architectures

6.1 Convolutional Neural Networks

6.2 Physics Informed Neural Networks

6.3 LSTM architecture and Capturing Time Series Data

7 General Statistical Tools

7.1 Hidden Markov Models

7.2 Monte Carlo Methods

7.3 Random Forests

8 Implementing Our Algorithm

8.1 Discussing our priorities

8.2 Full Algorithm